



Pengenalan Python

Alif Husnul Fikri



Programming

Membuat suatu set **perintah/instruksi** untuk mesin

- Analogi: memasak, memasang alat, terdapat instruksi untuk membuat alat

Memasak Mie

- Siapkan peralatan
- Nyalakan kompor dan siapkan panci
- Tambahkan air ke dalam panci sebanyak 500ml
- Masukkan mie

Instruksi harus jelas dan runut!



Algoritma

Kumpulan instruksi yang jelas dan terstruktur

Algoritma haruslah:

- Simple
- Lengkap
- Benar
- Tepat sasaran
- Efisien

Flowchart merupakan diagram yang dapat membantu kita membuat instruksi ke mesin dengan baik

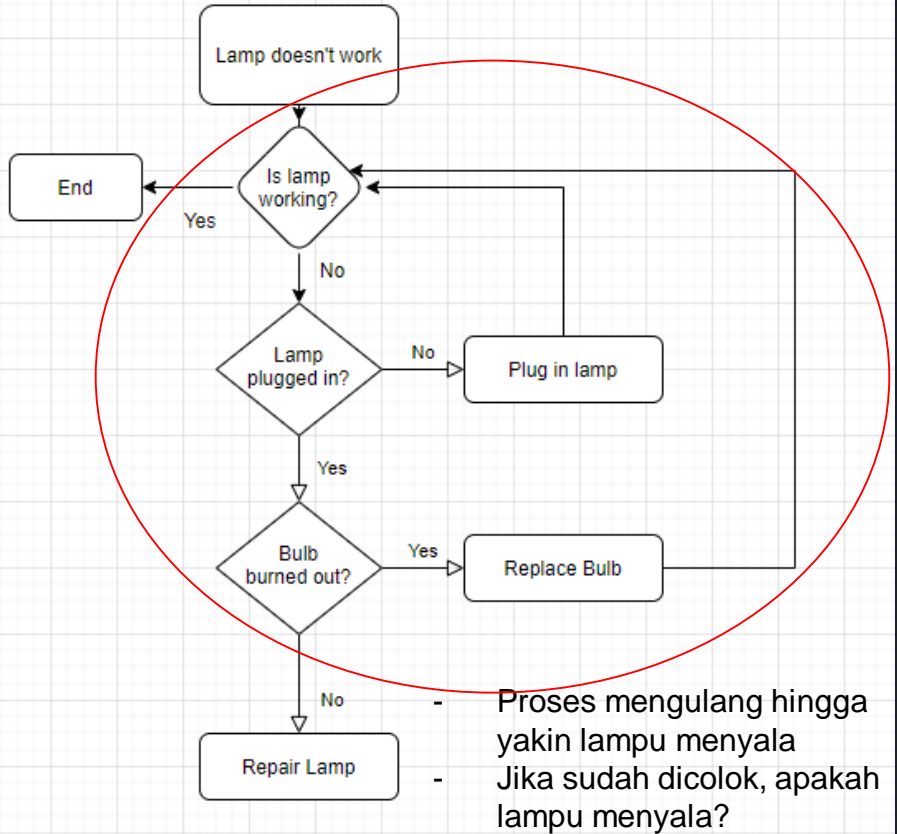
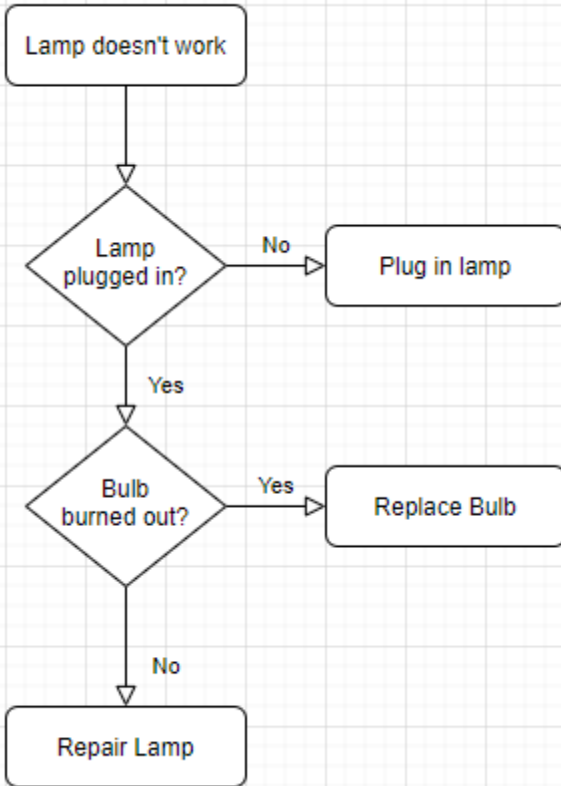
Proses jelas

Asumsi yang membaca instruksi tidak mengerti/pernah memasak mie



Proses tidak jelas

- Apakah panci perlu air atau tidak?
- Perlu dibuka bungkus mie nya? Atau masukan beserta bungkusnya?





Programming

Programming dapat menggunakan berbagai bahasa seperti:

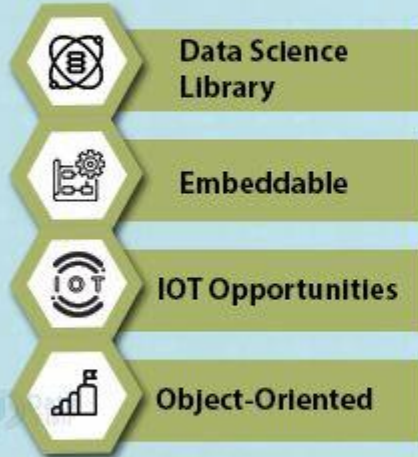
- Java
- PHP
- C++
- Python
- R

Tiap bahasa pemrograman memiliki **tata bahasa (syntax)** sendiri: layaknya bahasa inggris dan bahasa indonesia yang berbeda.

Tiap bahasa didesain untuk **lebih powerful** pada suatu bidang

Python

Python Advantages & Disadvantages



ADVANTAGES



DISADVANTAGES



Mana yang lebih mudah?

```
#include  
#include  
using namespace std;  
int main() {  
    string name;  
    cin >> name;  
    cout << "Good evening, " << name <<  
    endl;  
    return 0;  
}
```

```
name = input()  
  
print("Good evening, " + name)
```




Syntax, Code, Compiler

Merupakan aturan penulisan dari sebuah bahasa pemrograman (SPOK dalam bahasa Indonesia, imbuhan, dll).

Kumpulan syntax yang sudah disusun rapi disebut **code**. Semua tulisan/text yang dimulai dengan # merupakan komentar (bukan code)

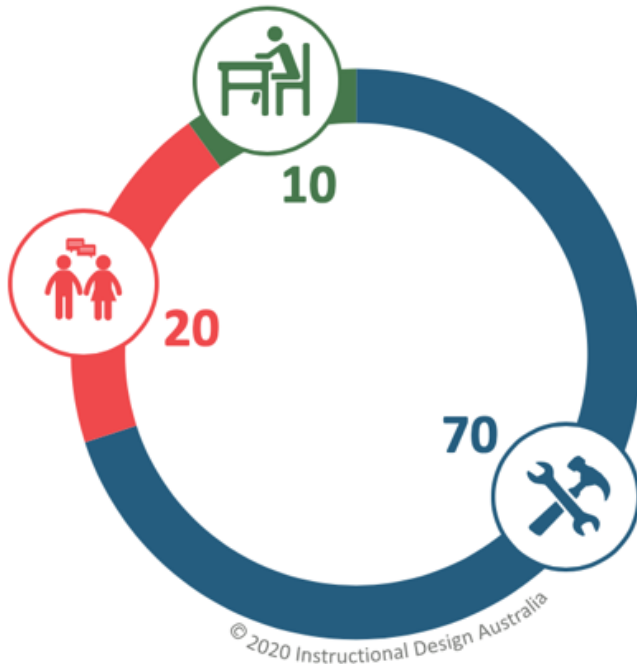
Code yang sudah siap dapat dijalankan, disebut sebagai **compile**

Code dapat dijalankan, menerima input/output melalui sebuah text box bernama **console** seperti **command prompt** atau bawaan dari **compiler**

Compiler -> Pycharm, spyder, anaconda, etc.

Latihan dan Explore

70:20:10 MODEL IN PRACTICE



EXPERIENTIAL

Applying learning on-the-job, situational learning in various contexts, experiences, challenges, practice.



SOCIAL

Coaching, mentoring, feedback, social learning, 'buddy' systems, sharing, collaboration.



FORMAL

Programs, courses, eLearning, simulations, reading.



Variable assignment

Dalam python, kita dapat **mendefinisikan** sebuah **variabel**

Nama simbolis untuk menyimpan sebuah **value**

Ditandai dengan **=**

Contoh :

- nama variable: a
- Value dalam a: 100

Nama variable biasanya ditulis agar mudah dimengerti

Beberapa aturan:

- Case sensitive, A dengan a berbeda
- Tidak boleh diawali angka/symbol kecuali _
- Tidak boleh menggunakan spasi, -, +, /, *

```
a = 100  
b = 200
```



Output

Output: Untuk mengeluarkan value ke console box (print)

```
print()
```

Output tidak memerlukan variabel karena kita tidak akan menyimpan value

Setiap print akan dipisahkan dengan satu baris enter

```
print("Selamat Pagi")
```

```
print(name)
```

- Jika **dalam kurung diberikan suatu value**, maka akan keluar output: **Selamat Pagi**
- Dapat berupa **text** ditandai dengan **kutip 1 atau 2 (' , ")**
- Dapat berupa **nama variable** yang akan menunjukan **value** dalam variable tersebut



Tipe Variable

Tipe	Penjelasan	Penanda	Contoh
string (str)	Text	" , '	"Selamat", 'selamat'
integer (int)	Bilangan Bulat		1, 2, 3
float (float)	Bilangan Desimal	. (angka setelah desimal)	1.5, 20.4, 0.5
boolean (bool)	Logika (Ya/Tidak)	True/False	True, False
None (none)	Kosong	None	None



Operasi Matematika

Terdapat syntax untuk melakukan operasi matematika dasar

Syntax	Penggunaan	Contoh
+	Penambahan	$100 + 200$, $a + b$, $\text{angka1} + \text{angka2}$, $a + 2$
-	Pengurangan	$500.2 - 200$, $a - b$, $\text{angka1} - \text{angka2}$, $a - 2$
*	Perkalian	$500 * 2$, $a * b$, $\text{angka1} * \text{angka2}$, $a * 2$
**	Pangkat	$2 ** 2$, $a ** b$, $a ** 2$, $a ** \frac{1}{2}$ (sama dengan akar a)
/	Pembagian	$500 / 2$, a / b , $a / b * c$
()	Prioritas	$(500 * 2) + 1$, $a * (b + c)$



Aturan matematika

Mengikuti aturan pada umumnya.

Angka/variable dalam kurung akan didahulukan

Urutan

- Dalam kurung
- Pangkat, kali, bagi
- Tambah, kurang

```
10 * 2 + 1 #hasil 21
10 * (2 + 1) #hasil 30
9 / 2 * 10 # 45
```



input

Input: untuk menerima input dari user melalui console box

```
name = input()
```

Variable, hasil yang **diinput** user akan **disimpan** dalam memori dengan nama ***name***

input -> merupakan **syntax** untuk menerima input dari **user**

Semua input yang diberikan user memiliki tipe string -> **perlu diubah**

```
name = input("Selamat Datang")
```



Jika diisi sesuatu, maka akan keluar output: **Selamat Datang**
Sebelum user melakukan input



Common Practice

Dalam programming, hal yang sering dilakukan sebelum membuat/menulis program adalah membuat flowchart/pseudocode

Tahap:

- Definisikan masalah yang ingin diselesaikan (**problem definition**)
- Buat **flowchart**
- Buat tulisan semi-kode yang menggabungkan syntax dasar dengan bahasa pada umumnya (**pseudocode**)
- Tulis program sesuai dengan aturannya (**code**)

Jika terjadi kesalahan pada penulisan atau ketidaksesuaian dengan aturan awal, maka program akan salah. Hal ini disebut sebagai **bug** -> harus dilakukan **debugging**



Contoh

Buat program untuk menerima input dari user lalu mengeluarkan input dari user!

Problem: program input dan output

Flowchart: mulai -> input dari user -> output dari user -> selesai

Pseudocode:

1. Variable A = input dari user
2. Print variable A

Code:

```
#program input dan output (code 0)
var_a = input()
print(var_a)
```



Contoh (alt 1)

Buat program untuk menerima input dari user lalu mengeluarkan input dari user!

Problem: program input dan output

Flowchart: mulai -> **output: “silakan masukan angka”** -> input dari user -> output dari user
-> selesai

Pseudocode:

1. Print “silakan masukan angka”
2. Variable A = input dari user
3. Print variable A

Code:

```
#program input dan output (code 1)
print("silakan masukan angka")
var_a = input()
print(var_a)
```

```
#program input dan output (code 2)
var_a = input("silakan masukan angka")
print(var_a)
```



Exercise

1. Mengalikan angka yang diinput user dengan 4
 - Problem: angka yang dimasukan user akan dikalikan 4
 - Instruksi:
 - input angka ke dalam variable
 - Angka * 4
 - Keluarkan angka
1. Menambahkan dua input user
 - a. Problem: menambahkan dua input dari user
 - b. Instruksi:
 - i. Masukan input 1
 - ii. Masukan input 2
 - iii. Hasil = Input 1 + input 2
 - iv. Keluarkan hasil



Warning

Dalam buku yang akan diberikan di drive, contoh menggunakan python 2.7. Sekarang, python 2.7 sudah tidak update dan digunakan.

Python yang terbaru: 3.8.x

Perbedaan mencolok python 2.7 dan python 3.8

- `print "Halo"` vs `print("Halo")`
- division, `3 / 2` akan menghasilkan 1 pada python 2.7, sedangkan pada python 3 akan menghasilkan 1.5



Tipe variable: multiple values

Terdapat tipe variabel yang dapat menyimpan data/values dalam satu variable.
Sebagai contoh -> **list**

```
simpsons = ['homer', 'marge', 'bart'] # -> list
```

List dapat menyimpan berbagai value dengan berbagai tipe, ditandai dengan **kurung siku** ([]) dengan setiap elemen dipisah oleh **koma** (,).

Ukuran list dapat diakses dengan **len()**

```
print(len(simpsons))  
panjang_list_simpsons = len(simpsons)  
print(panjang_list_simpsons)
```



Tipe variable: multiple values

Terdapat tipe variable yang dapat menyimpan berbagai macam value dalam sebuah variable

Tipe	Tuples	List	Set	Dict	String
Penanda	()	[]	set(), { }	{ }	" "
Contoh syntax	a = (1,2,3)	b = [1,2,3]	c = {1, 2, 3}	d = {'angka1' : 1, 'angka2': 2}	e = 'halo'
Key Point	Tidak dapat diubah/ditambah	Bisa diubah/ditambah	Hanya menyimpan unique value	Terdapat key: value, key harus unique	Menyimpan value berupa karakter
Mutable (dapat diubah?)	Tidak (immutable)	Bisa (mutable)	Bisa (mutable)	Bisa (mutable)	Tidak (immutable)
Penggunaan	Jika value tidak ingin diubah	Tipe variable yang lebih flexibel	Jika ingin menghilangkan duplicate value dalam list/tuples	Ingin menyimpan value seperti user_name dan password, user_name dan email	Text

Indexing

List memiliki values dan index. Value dalam sebuah list dapat diakses dengan syntax **indexing**

- **Indexing** ditandai dengan [] setelah nama variable, diisi dengan angka index yang diinginkan
- `simpsons[0]`

	← length = 5 →				
	'p'	'r'	'o'	'b'	'e'
index	0	1	2	3	4
negative index	-5	-4	-3	-2	-1

<https://www.programiz.com/python-programming/list>

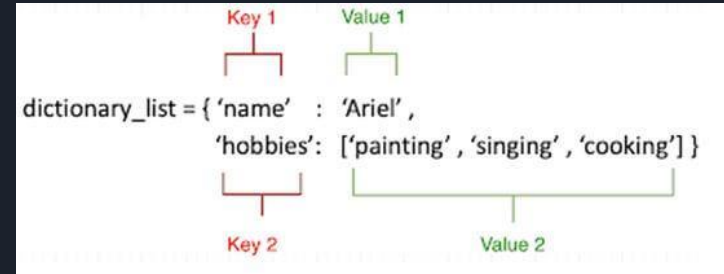
```
simpsons = ['homer', 'marge', 'bart']

# indexing untuk list, set, tuples
print(simpsons[0])
print(simpsons[2])
print(simpsons[-1])
```


Indexing (dict)

Dict memiliki cara sendiri untuk mengakses value di dalamnya

- **Indexing** ditandai dengan [] setelah nama variable, ditambah **key**
- `dictionary_list["name"]`



<https://www.gangboard.com/blog/python-dictionary>

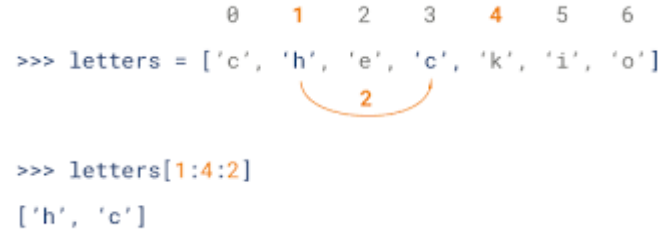
```
# indexing untuk dict
print(dictionary_list["name"]) #Ariel
print(dictionary_list["hobbies"]) # ['painting', 'singing', 'cooking']
```

Slicing (List, set, tuples)

Slicing (memotong) merupakan cara **indexing** untuk mendapatkan berbagai value dalam list

- ditandai dengan [] setelah nama variable, ditambah **nomor index awal, index akhir**
- `var_list[start:stop:step]`
- Start: Index awal, default 0
- Stop: Index akhir, default panjang list, index terakhir tidak akan dikeluarkan
- Step: Jumlah langkah, default 1

```
# indexing untuk dict
print(simpsons[0:2]) # mulai dr 0, berhenti di 2 (2 tidak termasuk)
print(simpsons[0:]) # mulai dr 0 hingga akhir
print(simpsons[:2]) # mulai dr awal hingga index 2
print(simpsons[1:3:2]) #1, 3 (tidak termasuk 3)
```



```
>>> letters = ['c', 'h', 'e', 'c', 'k', 'i', 'o']

>>> letters[1:4:2]
['h', 'c']
```

<https://py.checkio.org/blog/python-slices/>



Variable Methods

Dalam sebuah tipe variable, terdapat method yang dapat dipanggil untuk melakukan sesuatu.

```
simpsons = ['homer', 'marge', 'bart']  
  
# method  
simpsons.append('lisa') # menambahkan elemen ke dalam list  
print(simpsons) #['homer', 'marge', 'bart', lisa']
```

```
simpsons.append('lisa')
```

`simpsons` -> merupakan nama variable

`append()` -> merupakan method, ditandai dengan titik setelah variable. Kadang dapat menerima value, kadang tidak

`'lisa'` -> merupakan value yang diterima method

Beberapa Methods

Syntax	Penjelasan	
<code>lst.insert(i, x)</code>	Memasukan value x ke dalam index i dan menggeser ke kanan elemen setelah index i	<code>lst.insert(1, 3)</code>
<code>lst.clear()</code>	Menghilangkan semua elemen di dalam list, tidak mengeluarkan output	
<code>lst.copy()</code>	Mengcopy list, tidak mengeluarkan output	
<code>lst.count(x)</code>	Menghitung jumlah value di elemen list yang memiliki nilai x, mengeluarkan output	<code>lst.count(2)</code>
<code>lst.extend(iter)</code>	Memasukan <i>iter</i> variable ke dalam list (misal list, set, tuple)	<code>lst.extend([1,2,3])</code>
<code>lst.index(x)</code>	Mencari index pertama dalam list dengan value elemen sama dengan x, mengeluarkan output	<code>lst.index(100)</code>
<code>lst.sort()</code>	Mengurutkan list, tidak mengeluarkan output	
<code>lst.remove(x)</code>	Menghilangkan elemen pertama dalam list yang memiliki value sama dengan x, tidak mengeluarkan output	<code>lst.remove(x)</code>
<code>lst.reverse()</code>	Membalikan urutan list, tidak mengeluarkan output	

Beberapa method dalam dict

```
# create a dictionary (two ways)
family = {'dad':'homer', 'mom':'marge', 'size':6}
family = dict(dad='homer', mom='marge', size=6)

# convert a list of tuples into a dictionary
list_of_tuples = [('dad','homer'), ('mom','marge'), ('size', 6)]
family = dict(list_of_tuples)

# examine a dictionary
family['dad']      # returns 'homer'
len(family)        # returns 3
family.keys()      # returns list: ['dad', 'mom', 'size']
family.values()    # returns list: ['homer', 'marge', 6]
family.items()     # returns list of tuples:
                  #   [('dad', 'homer'), ('mom', 'marge'), ('size', 6)]
'mom' in family    # returns True
'marge' in family  # returns False (only checks keys)

# modify a dictionary (does not return the dictionary)
family['cat'] = 'snowball'      # add a new entry
family['cat'] = 'snowball ii'   # edit an existing entry
del family['cat']               # delete an entry
family['kids'] = ['bart', 'lisa'] # value can be a list
family.pop('dad')               # removes an entry and returns the value (
                                #   'homer')
family.update({'baby':'maggie', 'grandpa':'abe'}) # add multiple entries

# accessing values more safely with 'get'
family['mom']                  # returns 'marge'
family.get('mom')              # same thing
```



Sets

```
# create a set
languages = {'python', 'r', 'java'}           # create a set directly
snakes = set(['cobra', 'viper', 'python'])    # create a set from a list

# examine a set
len(languages)                                # returns 3
'python' in languages                          # returns True

# set operations
languages & snakes                             # returns intersection: {'python'}
languages | snakes                             # returns union: {'cobra', 'r', 'java', 'viper', 'python'}
languages - snakes                             # returns set difference: {'r', 'java'}
snakes - languages                             # returns set difference: {'cobra', 'viper'}
```



Range

Range digunakan untuk membuat urutan angka dari suatu nilai ke nilai lainnya.

`range(start, end, step)`

- syntax umum
- start: angka awal
- end: angka akhir (tidak termasuk)
- step : langkah atau step urutan

`range(end)` -> jika ditulis seperti, start value akan diisi nilai default = 0, dan step dengan value 1

`range(start, end)` -> jika ditulis seperti, start value akan diisi nilai default = 0, dan step dengan value 1

Untuk memasukan ke dalam list, dapat menggunakan `var_a = list(range(0, 100, 5))`



Exercise

1. Buatlah list dari 1 - 1000 loncat 50
2. Buatlah list dari 10 ke 1
3. Buatlah list dari 0 ke 10
4. Buatlah dict berisi:
 - a. Nama: Andi
 - b. Kelas: 12
 - c. Umur: 19
 - d. Berat Badan: 90
 - e. Tinggi: 170
 - f. Buat keys yang bernama BMI, dengan perhitungan $BMI = \text{berat_badan} / \text{tinggi_badan (m)}^2$